AMIGA

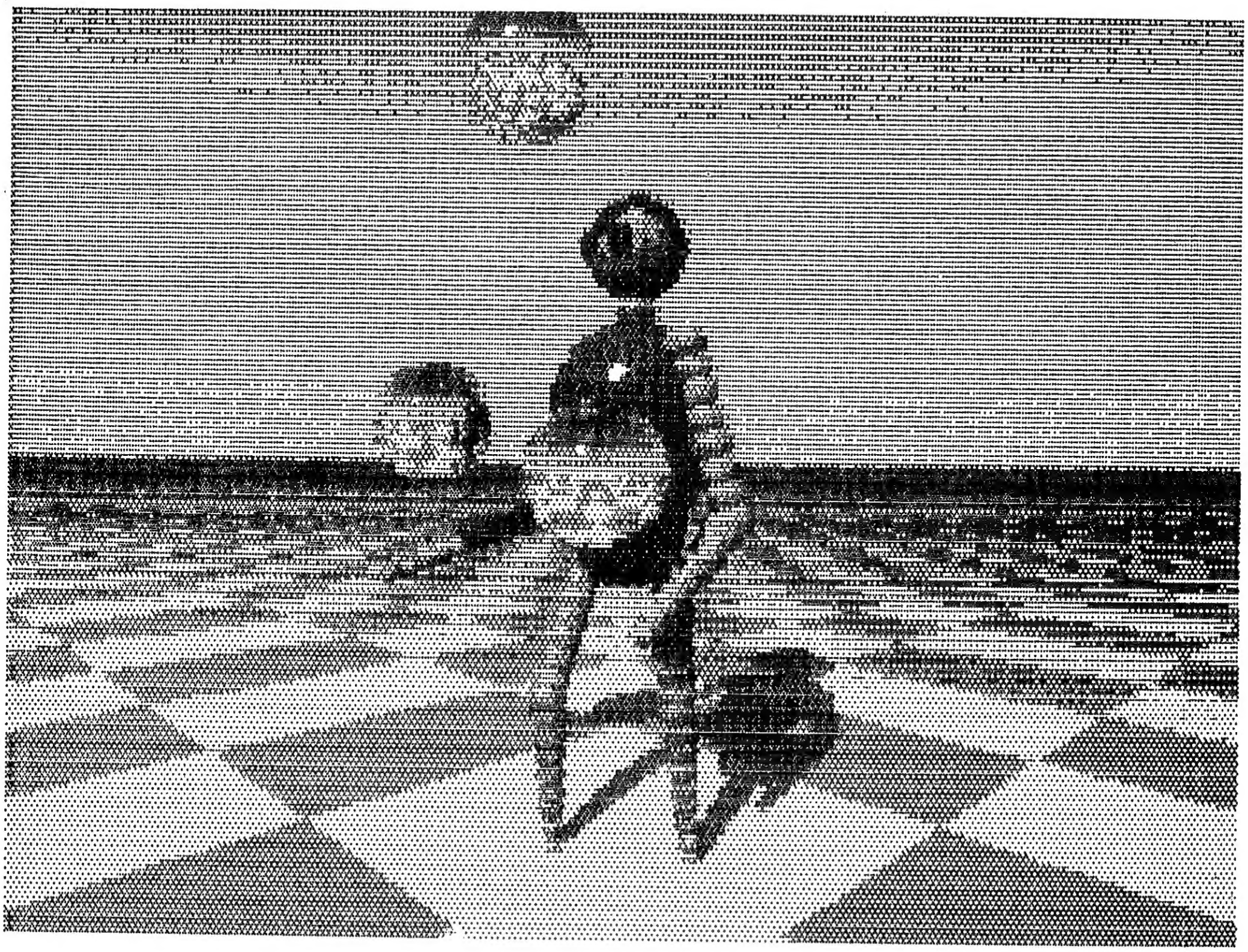
WORKBENCH

FOR THE COMMODORE AMIGA USER

$2
RRP

Registered by Australia Post – Publication No. VBG7930

Number 9 | Circulation: 500 | February 1987



Next Meeting

Sunday, February 8th, 1987 at 2pm

AUG meetings are held at Victoria College, Burwood Campus
in Lecture Theatre 2. Melways map 61 reference B5.

Amiga Users Group, PO Box 48, Boronia, 3155, Victoria, Australia

AMIGA™ Users Group

P.O. Box 48, Boronia, 3155, Victoria, Australia

Amiga Users Group

The **Amiga Users Group** is a non-profit, self-help group, made up of people interested in the Amiga computer and related topics.

Club Meetings

Club meetings are held at 2pm on the second Sunday of each month at Victoria College, Burwood Campus, in Lecture Theatre 2. Details on how to get there are on the back cover of this newsletter. The dates of the next few meetings are:

Sunday, February 8th at 2pm
Sunday, March 8th at 2pm
Sunday, April 12th at 2pm

Production Credits

This month's **Amiga Workbench** was edited by Peter Jetson. Equipment and software used was: TurboDOS S-100 computer, Diablo 630 printer, Gemini 10x printer, Wordstar and Fancy Font. Now that Desktop Publishing software for the Amiga is becoming available, we might soon be able to do the whole newsletter on an Amiga.

Copyright and Reprint Privileges

Contributions

Articles, papers, letters, drawings and cartoons are actively sought for publication in **Amiga Workbench**. It would be appreciated if contributions were submitted on disk, since that means they don't have to be re-typed! We have access to a wide range of computers, so we should be able to accept almost any type of disk, but Amiga disks are certainly the easiest. All disks will be returned! Absolute deadline for articles is the last weekend of the month before the cover date. Contributions can be sent to:

The Editor, AUG, PO Box 48, Boronia, 3155
(Note the new address)

AUG Users Group Disks

Disks from the **Amiga Users Group** Library are available on quality 3.5" disks for $10 each including postage on AUG supplied disks, or $2 each on your own disks. The group currently holds 58 public domain volumes, sourced from the USA, with more on the way each month.

Member's Discounts

The **Amiga Users Group** is currently negotiating discounts for its members on hardware, software and books. Members will be notified when negotiations are complete.

Currently, **Technical Books** in Swanston Street in the city offers **AUG** members a 10% discount on computer related books, as does **McGills** in Elizabeth Street. Just show your membership card. Although we have no formal arrangements with other companies yet, most seem willing to offer a discount to AUG members. It always pays to ask!

Membership and Subscriptions

Membership of the **Amiga Users Group** is available for an annual fee of $20. To become a member of AUG, fill in the membership form in this issue (or a photocopy of it), and send it with a cheque for $20 to:

Amiga Users Group, PO Box 48, Boronia, 3155

Amiga Users Group Committee

John Holland . . .	(Co-ordinator) . . .	348 1358	Carlton
Bob Scarfe	(Vice Co-ordinator) .	376 4143	Kensington
Ron Wail	(Meeting chairman) .	878 8428	Blackburn
Eric Salter . . .	(Secretary)	861 9117	Kew
Paul Radford . . .	(Treasurer)	663 3951	BH only
Neil Murray . . .	(Membership)	792 9666	Dandenong
Stephen Thomas . .	(Membership)	830 5783	Canterbury
Peter Jetson . . .	(Newsletter editor) .	762 1386	Boronia
Geoff Sheil . . .	(Assoc editor) . . .	509 3151	Armadale
Jo Santamaria . .	(Assoc editor) . . .	836 9129	Canterbury
Ron Van Schyndel .	(Librarian- books) .	882 7264	Hawthorn
Geoff Wood	(Librarian- books) .	580 7463	Aspendale
Ron Wail	(Librarian- software)	878 8428	Blackburn
Mike Creek	(Librarian- software)	878 9039	Blackburn
Bohdan Ferens . .	(Purchasing)	792 1138	Dandenong
Justin Somers . .	(Without portfolio) .	553 0379	Moorabbin

When phoning committee members, please try to be a bit considerate and not call at meal-times, late at night, or during popular TV programs. If you only have a general query, try to ring the member who lives closest to you.

Have you noticed our change of address?

Defenders of the Crown, a review by Fergus Bailey

I would like to start by saying that I do not really enjoy computer games. Sure, I occasionally crank up Hitchhickers Guide To The Galaxy, or more lately Leather Goddesses of Phobos, but on the whole I don't spend much time on games.

Most games to date for the Amiga have been directly ported from lower-powered computers and have been very disappointing considering the Amiga's capabilities. I waited with anticipation for the release of **Marble Madness**, and was pleased to see a game that started to do justice to the AMIGA, though I was a little disappointed that there were only six levels in the game. Surely, I thought, with the great graphics and sound laid on with the Amiga, there must be potential for a game that would even get a play-a-computer-game-once-in-a-blue-moon type like me interested.

Finally that game is here. It's a "Cinemaware Production, an interactive movie" from Mindscape, by the title of Defenders of the Crown. On booting up the disks (Yes, Virginia, there are **two** of them), you are greeted with credits, an opening screen which sets the high standard that is maintained throughout the game, and a cast of characters whom you will probably meet as you play. After being told the story so far, you find yourself in Sherwood Forest talking to Robin Hood, who tells you that you have the job of driving the Normans from England (You will play the part of a Saxon). Robin pledges you his help three times during the campaign, if you ask for help more than three times in one game, Robin will politely tell you that you're on your own.

The most outstanding thing about Defenders is the high quality graphics. Jim Sacks is the Art Director and the art is just mind blowing (Sacks' art was on the first slide show I ever saw for the Amiga and his work is still the best free hand Amiga pictures I have seen). The attention to detail is fantastic, and the whole production is a great compliment to the entire art team.

Defenders seems to have something for everyone; it includes strategy, adventure and arcade-action with top notch graphics and a good quality story line. Without telling you too much about the story, Defenders lets you go raiding, go jousting, rescue maidens and invade other territories. The object of the game is to end up in control of England by driving the Normans from the country. Although this may not sound like much, there are five other lords, all of whom are also interested in ruling England, and believe me, doing battle with an opponent like Edmund the Grim or Wolfric the Wild is no easy task.

As mentioned above, Defenders comes with two disks and is best played on a twin-drive system. The game is disk-intensive and although I had been told that Defenders could be played on a single drive Amiga, I was doubtful that this would work very well. To prove the point, I played a game using just my internal drive and was surprised to find that the game played quite well. In around forty minutes of play, I had to swap disks fourteen times. I think that Defenders would run best with a two meg. RAM expansion board but alas, these are so expensive that not many Amiga owners will get a chance to try this.

Despite all the great features, I still have a few complaints about Defenders, the main one being that there is no joystick input. Every time you go raiding, the sword fight with your opponent results in you bashing the daylights out of your mouse. A far better idea would have been to use the mouse for movement and the second port with a joystick for the action sequences. I'm sure that mouse manufacturers will be laughing all the way to the bank over this "feature". My other complaint is the lack of a game-save option; this is a bit of a nuisance, though with a program this big it would have been a bit of a problem. It's not a major concern, however, since a game only lasts about an hour or so before you either lose your castle or win control of England.

Defenders of the Crown is the best game I have seen for the Amiga and I would highly recommend it to anyone with an interest in games.

-- Fergus Bailey

Superbase Personal

My copy of **Superbase Personal** arrived only a fortnight ago, and I am already using it. It is important to be aware that I come to this from the perspective of a busy Parish Minister, so I do not have a great deal of time to be getting into why and how the thing works, and neither am I a programmer, so the thing has to be simple and easy to use.

Right from the outset, I found **Superbase Personal** a delight. Like all previous software from Precision Software, it is well packaged and well presented. Precision wrote **Easy Script** (word processor) and **Superbase 64** (database) for the Commodore 64, and I speak with considerable experience and pleasure of these programs.

My initial impression of **Superbase Personal** is that it will more than fulfil my expectations.

What you get for your money is a disk with the program on it, a manual and a "dongle" (a copy protection device).

The manual is clear and simple with well layed-out and helpful tutorials, but with one annoying feature: the type-style used in the manual has a peculiar "I" which makes it hard to distinguish from the letter "l".

Superbase makes it easy to set up files, records and fields and to move around on the screen using the mouse. In fact, the whole system can be operated by mouse. The program is menu driven and allows you to very quickly re-arrange your data and print it or display it or even put it into another file simply by clicking on the various gadgets and requesters. The data is stored in a way that enables the system to very quickly find anything you want. The heart of **Superbase** is the 'filter' which enables you to 'home-in' on precisely the piece of information you want. I have already used it to set up various operations for which I was using a spreadsheet, and such is the ease of use that a complete novice could operate it within a few minutes.

Superbase Personal is an excellent program, and a worthy successor to **Superbase 64**. I believe that Precision are soon to offer an upgrade called **Superbase Professional**, which will include a word-processor. If it is as good as the combination of **Superbase 64** and **Easy Script**, then I'll junk **Textcraft** in favour of the upgrade as fast as I can! One of the key applications for which I need a word-processor is to send out form letters into which I can merge files and data from a database and I presently find that without the mail-merge function I am severely limited.

I would be happy to demonstrate what I have learned about how to use **Superbase** to anyone interested.

-- David Peel

I've also got a few questions. Is there anyone who knows of a program which will enable my wife to write musical scores using the Amiga? We don't want the sort of "mickey-mouse" scores which **The Music Studio** produces, but properly written scores where notes can be joined together and key signatures changed in the same line and all that technical music stuff. We don't need it to sound or play the music. I guess what we are looking for is a sort of a "word processor" for music. Is there anyone out there who can help?

I wrote this review using Textcraft, and if I sound a bit frustrated with it, that is a fair assessment. Is there anyone who knows how to open a document without having to load the whole directory each time? I want to know how to load a specific file, so if anyone knows how, please tell me!

-- David Peel

Introduction to AmigaDOS, Part 2

In my last article, we looked at AmigaDOS, the operating system of the Amiga, and how it fits in with Workbench and the CLI. We also looked briefly at both the CLI and Workbench, outlining what their functions were and the relationship between them. This month, we will look more closely at the CLI, and find out what it is and how much power it gives the user. Hopefully, you will begin to use the CLI and discover the real Amiga and its capabilities.

The CLI, or command line interface, reads text from a device (which is usually the keyboard, but it may be something else) and attempts to load the program that was named on the command line. If the program was loaded correctly, the CLI allows it to begin executing (running). If it was not loaded properly, it says so with an error message! e.g."error 121: file is not an object module" or "103: insufficient free storage" or some other cryptic message (but at least it's better than most other operating systems). For example:

```
1>dir
```

When you type this command, the CLI sends requests to AmigaDOS to find the file "dir". When it is found, further requests are issued to bring the file off disk and into memory. When this is completed, the CLI relinquishes control and goes to "sleep", and allows the program "dir" to run. "dir" will then display a directory listing on the screen. This process is virtually identical every time you type a command at the "1>" prompt. This is a very simplified outline of what actually takes place, but is essentially correct, and any more detail serves only to confuse at the moment.

Let's look at some of the commands that are available to you on the "Workbench" disk of your Amiga. They are found in the "c" directory of your disk. As "Workbench" comes, it is configured to start up in the Workbench iconic interface. To enable you to use the CLI, you must do one of two things: 1. You must enter the preferences program and select the on position for the CLI on/off switch. You must then save this configuration. Now when you open the system drawer, you will find another icon has appeared called CLI. You may now open it up and you have a CLI window to play with. 2. If you want your Amiga to start up in the CLI when it first comes on, you will have to edit the file "startup-sequence" found in the "s" directory so that "Workbench" is not loaded and the initial CLI is not closed.

DIR - The Directory Command

This is the single most used command on any computer. It allows you to see the files on your disk. If you type "dir", it will display the names of files and directories in the current directory. The concept of "current directory" is an easy one. If you think of an inverted tree with the root directory being at the top and branches representing directories with files hanging from these branches, you understand the hierachical directory structure of AmigaDOS. Now the "current directory" can be set with the "CD" command and is like setting your pointer to one of the branches. Now the DIR command will only display files or directories at that level. The DIR command has several options, the most useful one is "opt a" which will let you see all files and directories from your current directory down. If CD is set to the root directory, then an "opt a" will display the whole disk contents. Some examples:

```
Dir       - lists the files & Directories in the CD
Dir opt a - lists all files & Directories from the CD
Dir DF0:  - list the files etc on drive DF0
Dir DF1:  - as above but on drive DF1
Dir >"PRT:" opt a  - re-directs output to the printer
                     instead of the screen.
```

> & < - Re-direction commands

The UNIX(tm)-like > & < re-direction directives are used to alter which file the information is sent to or got from. The ">" tells the operating system to re-direct output to the file mentioned after the symbol. The file may be a disk file or a device (like the printer) or anything else that will accept output from the computer. Similarly the "<" symbol, allows a program to accept its input from a file other than the Keyboard e.g. the external serial port or a disk file.

Useful commands - Diskcopy, Copy, Format

Diskcopy allow you to copy one disk to another. It will not allow you to copy "copy protected" disks however. It is invoked simply:

```
Diskcopy DF0: to DF0:
```

This will copy one disk to another using DF0. Of course you will have to swap disks a minimum of 3 times with this command. If you have a second (external) drive you could type Diskcopy df0: to df1: and no swaps would be necessary.

Copy allows single files to be transferred from one place to another. E.g.

```
Copy DF0:foo to DF1:
```

which will copy the file called "foo" from DF0 to DF1. You can copy from device to device, too.

```
Copy DF0:foo to RAM:
```

will copy "foo" to the RAMdisk device.

Format enables us to format a blank disk for use and give the disk a name. eg.

```
Format drive DF0: name Foo
```

will format the disk in drive DF0 and name it "Foo". It will be empty, but we can now copy files to it for storage.

The various commands in the "c" directory are useful in different situations. Some will probably never be used by the average user. For a full explaination of the commands, a book like the "AmigaDOS Manual" by Bantam is essential. If you are serious about learning to use the CLI then get one - they are worth it.

In the next article, we will look at more of the commands and some features such as wildcards, a bit more on re-direction and look at how to create "execute" files and, if room permits, look at how the Amiga Terminal is used practically. But as a preamble to further articles, lets take a look at how AmigaDOS sees the terminal.

The Amiga Terminal - The Console Device

As we mentioned in the last article, the CLI is like the traditional interface that most computers have, for example, the "A>" prompt under MS-DOS on the IBM-PC and clones, the "1A>" prompt of CP/M and TurboDOS. They are all essentially the same, allowing the user to type commands to the computer using a terminal which is made up of a screen, a keyboard and some electronics, plugged into the computer. In the Amiga however, there is no terminal as such, but there is a program, called the console device, which makes the special chips and other hardware and software emulate a terminal. This terminal looks to your program (and you) to be an ANSI (American National Standards Institute) x3.64 standard terminal.

To most people, a terminal is a screen and keyboard that is separate from the computer, which physically resides somewhere else. The Amiga is different only in the fact that the terminal is part of the computer hardware and its functioning, intimately related to the functioning of the whole system (and therefore can't really be regarded as a separate entity). The CLI communicates with you via this Console device (terminal). The Console device then receives your input (from keyboard and mouse), from the input device and sends output to the screen via the layers library (which arbitrates between different programs that want to write to the screen, and essential in a multi-tasking environment).

The Input device itself uses the Keyboard and gameport devices for keyboard and mouse input respectively. So you see, it is all quite complicated, trying to emulate a terminal. However, this is not a real terminal but a "virtual" one. There can be up to 20 CLI's open (depending on available memory) and each independent! Each with its own window, and each window having a console device attached to it, acting like a full ANSI x3.64 terminal. Neat Huh! This is one manifestation of the multi-tasking (and potentially multi-user) environment of the Amiga.

An Introduction to Writing Your Own Device Drivers

by Alan Kent

Part 2

Writing a Device Driver

So now that we know what a device driver is, how do we go about writing one? In the RKM: libraries and devices appendix F a sample device driver written in assembler is shown. Looking at the code, I suspect it contains a few errors. Some pieces of code just didn't make sense to me. After looking at it for a while and (hopefully) understanding how it works, I threw it away and started again from scratch. The example in the manual is meant to be able to automatically load from disk. As I mentioned before, I did not worry about this as I wanted to get something working as quickly as possible and automatically loading C programs looked like it could be a bit of a problem (actually, one of the manuals stated it was not possible).

When writing a device driver, there are two main sections of code. The first section looks very much like a library - and in fact shares many structures and functions with libraries. This provides a nice consistent interface between the device driver and other programs that wish to use it. The second section is what actually performs all the commands.

First, a set of routines must be written which allow a device to be opened, closed and expunged (totally removed from memory). These are exactly the same as for a library. On top of these functions, two extra functions must be defined - BeginIO and AbortIO. BeginIO is a call which tries to immediately perform the IO operation without using the message port. If the operation cannot be done immediately, then the command must be queued by sending it to the message port. This allows simple status commands to be performed very quickly. The AbortIO command tries to abort an existing command.

One problem with writing a driver in C code is that Exec passes parameters for these calls in registers. This means that some pieces of assembler must be written to push the registers onto the stack before calling the C code. Note that as the operating system is calling functions written in C directly, if the C compiler requires special values in the registers then the code will not work. For example, I have heard that the Aztec C compiler uses a register to point to the global variables. If this is the case, then this register must be set up (probably by the same code that pushes the registers onto the stack) before the actual C function can be called.

Once these routines are written a device structure (struct Device) must be created using the MakeLibrary() call. MakeLibrary() accepts a pointer to an area of memory that defines how the library node is to be initialized (see the function InitStruct()). However it is very difficult to set up the necessary memory area for InitStruct() in C and so I simply initialize the device structure after calling MakeLibrary() using normal C code. Once this has been done, AddDevice() must be called to add the new device to the system. Once AddDevice() has been called, other programs may use all of the device function calls previously mentioned (OpenDevice() etc).

The example device driver in the manual creates a new process for each unit. Rather than trying to make more problems for myself by trying to create new processes from a C program, and as I only have a single unit anyway, I did not create a new process per unit but rather used the same process that creates and adds the device node to receive and process commands. So, before the device is actually added to the system, a port must be created to recieve commands. The Open code will put a pointer to this port into the IO request block which was passed in the OpenDevice() call. This is how DoIO() and SendIO() know where to send the message.

The Hard Disk Device Driver

The above discussion should be able to be used for any type of device driver you wish to write. I have been considering trying to write a printer spooler using device drivers too, although surely someone has done a printer spooler for the Amiga before (I had better go search through all the PD disks I guess).

The hard disk driver must emulate the trackdisk device driver in every way. Legal requests are defined in chapter 7 of the RKM: libraries and devices and include such commands as read data from disk, write data to disk, format a track, switch the motor on and so on. The commands are all fairly straight forward. The hard disk driver is much simpler than the floppy disk driver in many ways as the disk cannot be removed!

One area I found difficult to process was the sector labels. The trackdisk device actually allows an extra 16 bytes per sector to be written due to some sector header information on the disk. The hard disk controller I have does not allow this. As a result, I mark any commands that try to use this information as an error. So far, I have not found any code that actually uses this information so it is not a problem.

The trackdisk device driver (in V1.1 anyway) buffers a whole track of the disk in memory at a time. This buffering of data in memory is referred to as caching. V1.2 does better caching, but I dont know how (I dont have all the necessary documentation for V1.2). As a result, I tried to add some of my own more intelligent caching. Unfortunately, my caching seems to work quite well for about 5 minutes, but then it hangs the system. I will have to recheck the code sometime. It is difficult to determine however if the caching actually improves the speed of the hard disk or whether it runs slower due to the extra code that needs to be executed. The extra caching certainly uses up more memory, so at this stage I have removed all caching. Even without caching, its still faster than floppies.

My hard disk driver then consists of a number of sections. First, when it is loaded it creates and adds the device node to the system. It then waits for comands to arrive from a message port. When a command arrives, a switch statement decides what code to execute based on the command type. The disk read and write type commands map the offset and length fields in the command message onto head, track and sector numbers which is then fed to the hard disk controller. Other commands such as motor on/off and disk change count can be easily immitated (the motor can never be switched off and the disk change count is constant).

One area that can be confusing is that the code that is needed for handling open and close device calls made by programs making requests to the driver is never executed by the device driver process. When the device is added to the system, a set of pointers to these functions is put in the device structure allowing other tasks to execute code.

To install the driver involves adding a few lines to the S:STARTUP-SEQUENCE file. First the command RUN L:HARDDISK (where L:HARDDISK is my device driver program) starts up the device driver. Next the command MOUNT DH0: notifies AmigaDOS that the device can be used as a disk. For the mount command to work, the file DEVS:MOUNTLIST must have an entry for DH0: added. The file on my disk had an example entry for DF1: so I just copied it and changed the fields (such as number of cylinders, heads etc) for my hard disk. The name of the device driver also had to be changed from trackdisk.device to harddisk.device. I have also added to my S:STARTUP-SEQUENCE file some ASSIGN commands to change the C: directory to be on the hard disk. These changes need only be made once and then every time you boot up, you have a very big disk drive!

One problem I did have however was to make sure that the hard disk driver was actually loaded before I started using it. This is because the RUN command exits before the program to be run has even been loaded from disk. After the RUN command I put in a delay of about 5 seconds in the S:STARTUP-SEQUENCE file so that the driver should have time to load completely.

Hardware

This is a brief description of the hardware I have used. It does not auto-configure (oh dear, Commodore will be angry with me!) as I could not work out exactly how to do it. I have a copy of the expansion specs from Commodore (I sent off to the states) but the auto-configure was a bit beyond the effort I wanted to go to. I am not likely to add anything else to my Amiga anyway (famous last words). The controller card I used has 8 registers which I have mapped into memory space. Due to the 68000's 16 bit words, it ended up using 16 bytes of memory, the high bytes of the 16 bit words not being used. The controller's registers can be read from or written to and include cylinder number registers, status registers, command registers and so on.

Sending commands to the hard disk simply involves storing the relevant parameters in the controllers registers and sending a command such as read a sector, write a sector or format a track. All this is done by simple memory read and write commands. I ended up putting the controller card at the top of expansion memory at $9ffff0 (actually I do not decode the address to that precision, but its close). As long as I dont expand to 8 megs of memory, it should not interfere with anything - even other boards which do autoconfigure.

After sending the controller a command, the device driver must wait for the command to finish. I tried to use the interrupt line provided by the controller card, but at this stage I have not successfully got the interrupt handlers on the amiga to work. The approach I am currently using is to poll the status register of the controller. Polling is where the program sits in a loop continuously checking the status register until the command has finished. In order to give other tasks a chance to run while the driver is polling, I placed a call to Delay() inside the polling loop. The delay cannot be too long or else the disk will become too slow, but it does give other tasks a bit of a chance to run.

Looking at the side of the Amiga, the expansion bus is numbered as follows. Reading the hardware reference manual seems to say the numbering is different to this, but this is what I used (and it works). I would check the signal numbers before building this circuit in case of incorrect pin numbers due to typing errors.

```
1 3 5 7 9 ..... 85
==================      the edge of the board
2 4 6 8 10 ..... 86
```

WARNING: I do not take any responsibility if the following circuit contains any errors. All I can say is that it has not blown my amiga up yet. USE AT OWN RISK!

All numbers on the left are Amiga signals. All numbers on the right are WD-1002-05 controller signals. Parts should be LS or better for speed. All signals marked * are active when low (inverted). Wires that cross like this:

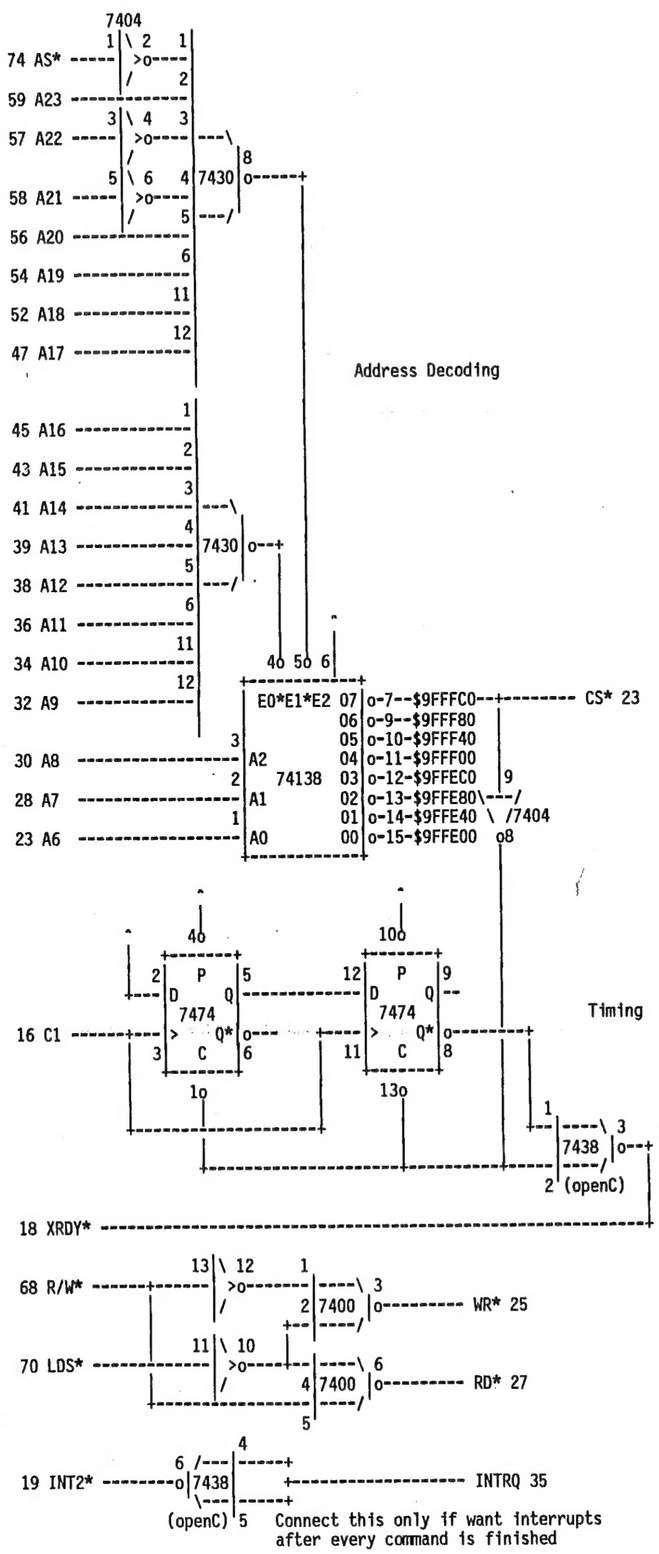
```
    |
----+----
    |
```

are not joined. Joins are shown by +'s as in this:

```
    |
----+----
    |
```

Pin Connections

```
21 A5 ------- (not used)
24 A4 ------- (not used)
26 A3 ------- RS2 21
27 A2 ------- RS1 19
29 A1 ------- RS0 17
53 RES* ----- MR* 39
75 PD0 ------ DAL0 1
77 PD1 ------ DAL1 3
79 PD2 ------ DAL2 5
81 PD3 ------ DAL3 7
83 PD4 ------ DAL4 9
86 PD5 ------ DAL5 11
84 PD6 ------ DAL6 13
82 PD7 ------ DAL7 15
```



A Way To ALIAS Commands

As you all know (don't you?) you can assign logical volume/device names to physical disk drive names such as:

```
assign Source: df0:
assign Destination: DF1:
diskcopy source: to destination:
```

Useful? Yes, for some things. Did you know you can also assign logical volume/device names to sub-directories? Such as:

```
assign from: df0:devs/printers
assign to: ram:
copy from: to to:
```

BUT the *FUN* one you may not have noticed is that you can assign to FILE NAMES (read commands). For example:

```
assign x: c:execute
assign e: c:ed
assign cc: df1:c/LC
```

then use them like:

```
x: startup-sequence
e: my text file
cc: foo -i df1:include
```

Assigns are SYSTEM WIDE, NOT unique to each window. Have fun!

AmigaDos Filename Wildcarding

As my contribution to figuring out those enigmatic commands in SYS:C, here's the way the LIST command handles "wildcards" in filenames.

If you want to see all the files with names ending in .c you type

```
    LIST PAT #?.c
or  LIST P   #?.c
```

All the files with names beginning with A: LIST P A#?

All the files with four-letter names: LIST P ????

<?> is the wild-card character, and <#?> means any number of <?>s. Comparisons ignore upper/lower case distinctions.

This should work on any command which lists the option "...P=PAT/K..." when you type the command name followed by a <?>: e.g. LIST ?

This and some other options which seem less useful than this are described quickly in:

"TRIPOS -- A Portable Operating System for Mini-computers"
Martin Richards, Aylward, Bond, Evans and Knight
Software Practice and Experience Vol 9 pp 513-526 (1979)

(TRIPOS is the OS on which AmigaDos is based)

There is a longer treatment of this topic in:

"A Compact Function for Regular Expression Pattern Matching"
Martin Richards
Software Practice and Experience Vol 9 pp 527-534 (1979)

Isn't it a pain to have to go get microfilmed articles to figure out how to use a computer you just bought?

To the Commodore-Amiga people: when WILL this stuff be available? Surely the Users Manual for the OS doesn't need to be restricted to developers?

Richard Muller
Computer Studies Program
Hampshire College
Amherst MA 01002

AmigaDos Tricks

The following tricks may be of interest to CLI/AmigaDos users.

1. First, you can move a file to another directory without performing a copy then delete. Use the RENAME command and specify the pathname as part of the destination string.

   For example:

   ```
   rename lottahelp newdir/lottahelp
   ```

   moves the file 'lottahelp' to another directory called 'newdir'.

   You can even give the file a different name. BUT-BUT-BUT, you can only move files from one device to another location ON THE SAME DEVICE. You cannot move a file from RAM: to DF1:, etc.

2. It is possible to set the system date/time by redirecting the stdin. But, you have to fool DATE by doing the following:

   ```
   DATE <newinfile [>nil:] ?
   ```

   The [] parameter is optional (do not included the '[' ']' symbols). It will redirect the output of the DATE command to the nil: device.

   The '?' is used to fool the date command into using 'newinfile' as the input for DATE.

Public Domain Update

At long last, the Amicus disks have finally arrived! There are a few major differences between these disks and the Fred Fish collection: these disks have a lot of programs in Basic, most of the programs are in source code only, and the disks contain icons so that almost everything is accessible from the Workbench.

My disk of the month is **Amicus #9.** This contains the **Boing!** demo program with selectable speed, as well as an Icon-based way to set the time and date. Also worth a look is **MyCLI,** a very interesting CLI replacement.

I'll let the rest of the volumes speak for themselves.

Amicus #1

ABasiC programs:	**Graphics**
3Dsolids	- 3D modeling program with sample data files
Blocks	draws blocks
Cubes	draws cubes
Durer	draws pictures in the style of Durer
FScape	draws fractal landscapes
Hidden	3D drawing program with hidden line removal
JPad	Simple paint program
Optical	draw several optical illusions
PaintBox	simple pain program
Shuttle	draws the shuttle in 3D wireframe
SpaceArt	graphics demo
Speaker	speech utility
Sphere	draws spheres
Spiral	draws colour spirals
ThreeDee	3D function plots
Topography	Artificial topography
Wheels	draws circle graphics
Xenos	draws fractal planet landscapes
ABasiC programs:	**Tools**
AddressBook	simple database program for addresses
CardFile	simple card file database program
Demo	multiwindow demo
KeyCodes	shows keycodes for a key you press
Menu	run many ABasiC programs from a menu
MoreColors	way to get more colours on the screen at once using aliasing
Shapes	simple color shape designer
SpeakIt	speech and narator demo
ABasiC programs:	**Games**
BrickOut	classic computer brick wall game
Othello	also known as "Reversi"
Saucer	simple shoot-em-up game
Spelling	simple talking spelling game
ToyBox	selectable graphics demo
ABasiC programs:	**Sounds**
Entertainer	plays that tune
HAL9000	pretends its a real computer
Police	simple police siren sound
SugarPlum	plays "The dance of the Sugar Plum Fairies"
C programs	
ATerm	simple terminal program SE
cc	aid to compiling with Lattice C
Decvnt	opposite of CONVERT for cross developers
Dotty	source code to the "dotty window" demo
EchoX	unix-style filename expansion, partial S+OD
FasterFP	explains use of fast-floating point math
FixDate	fixes future dates on all files on a disk SE
FreeDraw	simple Workbench drawing program SE
GfxMem	graphic memory-usage indicator SE
Grep	searches for a given string in a file ED
Ham	shows off the hold-and-modify method of color generation
IBM2Amiga	fast parallel cable transfer between an IBM and an Amiga
Mandel	Mandelbrot set program SE
Moire	patterned graphic demo SE
ObjFix	makes Lattice C object file symbols visible to Wack SE
Quick	quick sort strings routine
Raw	example sample window I/O
SetLace	turns on interlace mode SE
Sparks	qix-type graphic demo SE
Other executable programs	
SpeechToy	speech demonstration
WhichFont	displays all available fonts
Texts	
68020	describes 68020 speedup board from CSA
Aliases	explains use of ASSIGN command
CLICard	reference card for AmigaDOS CLI
CLICommands	guide to using the CLI
Commands	shorter guide to AmigaDOS CLI commands
EdCommands	guide to the ED editor
FileNames	AmigaDOS filename wildcard conventions
HalfBright	explains newer graphics chips that can do more colours
ModemPins	description of the serial port pinout
RAMDisks	tips on setting up your RAM: disk
ROMWack	tips on using ROMWack
Sounds	explanation of the Instrument demo sound file format
Speed	refutation of Amiga's CPU and custom chip speed
WackCmds	tips on using Wack

Amicus #2

C programs	
Alib	AmigaDOS object library manager SE
Ar	text file archiver program SE
FixObj	auto-chops executable files
Shell	simple CLI shell SE
Sq.Usq	file compression programs SE
YachtC	a familiar games SE
Make	a simple "make" programming utility SE
Emacs	early version of the Amiga text editor SED
Assembler programs	
bsearch.asm	binary search code
qsort.asm	qsort() function, source and C test program
setjmp.asm	setjmp() code for Lattice 3.02
SVprintf	Unix system V compatible printf()
trees.o	Unix compatible tree() function OD
John Draper Amiga Tutorial	
Animate	describes animation algorithm
Gadgets	tutorial on gadgets
menus	learn about intuition menus

Amicus #3

C programs	
Xref	a C cross-reference generator SE
6BitColor	extra half-bright chip gfx demo SE
Chop	truncate (chop) files down to size SE
CleanUp	removes strange characters from text files
CR2LF	converts CRs to LFs in Amiga files SE
Error	adds compile errors to a C file S
Hello	windows example from the RKM S
Kermit	generic Kermit implementation, flakey, no terminal mode SE
Scales	sound demo plays scales SE
SkewB	Rubik cube demo in hi-res colours SE
AmigaBasic programs	
Automata	cellular automata simulation
CrazyEights	card game
Graph	function graphing programs,
WitchingHour	a game
ABasiC programs	
Casino	games of poker, blackjack, dice and craps
Gomoku	a game
Sabotage	sort of an adventure game
Other executable programs	
Disassem	a 68000 disassembler ED
DpSlide	shows a given set of IFF pictures ED
Arrange	a text formatting program ED
Assembler programs	
ArgoTerm	a terminal program with speech output and xmodem SE

Amicus #4

Files from the original Amiga Technical BBS. Note that some of these files are old, and refer to older versions of the operating system. These files came from the Sun system that served as Amiga Technical Support HQ for most of 1985. These files do not carry a warranty, and are for educational purposes only. Of course, that doesn't necessarily mean they don't work.

Complete and nearly up-to-date C source to 'image-ed', an early version of the Icon Editor. This is a little flakey, but compiles and runs.

An intuition demo, in full C source, including files: demomenu.c, demomenu2.c, demoreq.c, getascii.c, idemo.c, idemo.guide, idemo.make, idemoall.h, nodos.c and txwrite.c

addmem.c	add external memory to the system
bobtest.c	example of BOB use
consoleIO.c	console IO example
creaport.c	create and delete port
creastdi.c	create standard IO requests
creatask.c	creating task examples
diskio.c	example of track read and write
dotty.c	source to the "dotty window" demo
dualplay.c	dual playfield example
flood.c	flood fill example
freemap.c	old version of "freemap"
geltools.c	tools for Vsprites and BOBs
gfxmem.c	graphics memory usage indicator
hello.c	window example from RKM
inputdev.c	adding an input handler to the input stream
joystik.c	reading the joystick
keybd.c	direct keyboard reading
layertes.c	layers example
mousport.c	test mouse port
ownlib.c, ownlib.asm	example of making your own library with Lattice
paratest.c	tests parallel port commands
seritest.c	tests serial port commands
serisamp.c	example of serial port use
prinintr.c	sample printer interface code
prtbase.h	printer device definitions
regintes.c	region test program
setlace.c	source to interlace on/off program
setparallel.c	set the attributes of the parallel port
setserial.c	set the attributes (parity, data bits) of the serial port
singplay.c	single playfield example
speechtoy,c	source to narrator and phonetics demo
timedely.c	simple timer demo
timer.c	exec support timer functions
timrstuf.c	more exec support timer functions
WhichFont.c	loads and displays all the available system fonts
process.i, prtbase.i	assembler include files
Texts	
autorqstr.txt	warnings of deadlocks with autorequesters
consoleIO.txt	copy of the RKM console IO chapter
diskfont.txt	warning of disk font loading bug
fullfcnt.txt	list of #defines, macros, functions
inputdev.txt	preliminary copy of the input device chapter
license	information on Workbench distribution license
printer	pre-release copy of the chapter on printer drivers from RKM 1.1
v11fd.txt	"diff" of .fd file changes from version 1.0 to 1.1
v28v1.diff	"diff" of include file changes from version 28 to 1.0

Amicus #5

Files from the Amiga Link/Amiga Information Network. Note that some of these files are old, and refer to older versions of the operating system. These files are from Amiga Link. For a time, Commodore supported Amiga Link, aka AIN, for online developer techincal support. It was only up and running for several weeks. These files do not carry a warranty, and are for educational purposes only. Of course, that's not to say they don't work.

C programs	
whereis.c	find a file, searching all subdirectories
bobtest.c	BOB programming example
sweep.c	sound synthesis example
Assembler programs	
mydev.asm	sample device driver
mylib.asm	sample library example
mylib.i, mydev.i, asmsupp.i, macros.i	assembler include files
Texts	
amigatricks	tips on CLI commands
extdisk	external disk specification
gameport	game port specification
parallel	parallel port specification
serial	serial port specification
v1.1update	list of new features in version 1.1
v1.1h.txt	"diff" of include file changes from version 1.0 to 1.1

Files for building your own printer drivers, including dospecial.c, epsondata.c, init.asm, printer.c, printer.link, printertag.asm, render.c, and wait.asm. This disk also contains a number of files describing the IFF specification. These are not the latest and greatest files, but remain here for historical purposes. They include text files and C source examples. The lastest IFF specification is elsewhere in this library.

Amicus #6

IFF Pictures: This disk includes the DPSlide program, which can view a given series of IFF pictures, and the "showpic" program, which can view each file at the click of an icon, and the "saveilbm" program, to turn any screen into an IFF picture. The pictures include a screen from ArticFox, a Degas dancer, the guys at Electronic Arts, a gorilla, horses, King Tut, a lighthouse, a screen from Marble Madness, the Bugs Bunny martian, a still from an old movie, the Dire Straits moving company, a screen from Pinball Construction Set, a TV Newscaster, the Paint Can, a world map, a Porsche, a shuttle mission patch, a tyrannosaurus rex, a planet view, a VISA card, and a ten-speed bike.

Amicus #7

DigiView HAM demo picture disk: This disk has pictures from the Digi-View hold-and-modify video digitiser. It includes the ladies with pencils and lollypops, the young girl, the bulldozer, the horse and buggy, the Byte cover, the dictionary page, the robot and robert. This includes a program to view each picture separately, and all together as separate, slidable pictures.

Amicus #8

C programs	
Browse	view text files on a disk using menus SED
Crunch	removes comments and white space from C files
IconExec	execute CLI commands from Workbench SE
PDScreenDump	dumps rastport of highest screen to printer
SetAlternate	sets a second image for an icon, when clicked SE
SetWindow	makes windows for a CLI program to run under Workbench SE
SmallClock	a small digital clock that sits in a window menu bar
Scrimper	the screen printer in the fourth Amazing Computing SE
AmigaBasic programs	(Note: Many of these programs are present on Amicus disk #1. Several of these were converted to AmigaBasic, and are included here)
AddressBook	a simple address book database
Ball	draws a ball
Cload	program to convert CompuServe hex files to binary
Clue	the game, intuition driven
ColorArt	art drawing program
DeluxeDraw	drawing program in the third issue of Amazing Computing SD
Eliza	conversational computer program
Othello	the game
RatMaze	3D ratmaze game
ROR	boggling graphics demo
Shuttle	draws 3D pictures of the space shuttle
Spelling	simple spelling program
YoYo	wierd zero-gravity yo-yo demo, tracks yo-yo to the mouse
Executable programs	
3Dcube	Modula-2 demo of a rotating cube
Alticon	sets a second icon image, displayed when the icon is clicked
AmigaSpell	a slow but simple spelling checker ED
Arc	the ARC file compression utility, must-have for telecommunicating ED
Bertrand	graphics demo
DiskSalvage	a program to rescue trashed disks ED
KwikCopy	a quick but nasty disk copy program, ignores errors ED

LibDir	lists hunks in an object file ED
SaveILBM	saves any screen as an IFF picture
ScreenDump	shareware screen dump program, E only
StarTerm	terminal program, xmodem, version 2 ED
Texts	
LatticeMain	tips on fixing main.c in Lattice
GDiskDrive	make your own 5.25 drive
GuruMed	explains the Guru numbers
Lat3.03bugs	bug list of Lattice C version 3.03
MForgeRev	user's view of MicroForge hard disk drive
PrintSpooler	execute-based print spooling program
.BMAP files	
	These are the necessary links between AmigaBasic and the system libraries. To take advantage of the Amiga's capabilities in Basic, you need these files. BMAPs are included for 'clist', 'console', 'diskfont', 'exec', 'icon', 'intuition', 'layers', 'mathffp', 'mathieeedoubas', mathieeesingbas', 'mathtrans', 'potgo', 'timer' and 'translator'.

Amicus #9

AmigaBasic programs	
FlightSim	simple flight simulator program
HuePalette	explains Hue, Saturation, and Intensity
Requester	example of doing requesters from Amiga Basic
ScrollDemo	demonstrates scrolling capabilities
Synthesiser	sound program
WorldMap	Draws a map of the world
Executable programs	
Boing!	latest Boing! demo, with selectable speed E
Brush2C	converts an IFF brush to C data instructions, initialisation code E
Brush2icon	converts IFF brush to an icon E
Dazzle	graphics demo, tracks to mouse
DeciGEL	assembler program for stopping 68010 errors
Klock	menu-bar clock and date display E
life	the game of life E
TimeSet	Intuition-based way to set the time and date
EMEmacs	another EMACS, more oriented to word-processing SED
MyCLI	a CLI shell, works without the workbench SED
Texts	
FnctnKeys	explains how to read function keys from AmigaBasic
HackerSin	explains how to win the game 'hacker'
1st68010	guide to installing a 68010 in your Amiga
PrinterTip	tips on sending escape sequences to your printer
StartupTip	tips on setting up your startup-sequence file
XfrmrReview	list of programs that work with transformer
PrinterDrivers	
	Printer drivers for: Canon PJ-1080A, C Itoh Prowriter, an improved Epson driver that eliminates streaking, Gemini Star-10, NEC 8025A, Okidata ML-92, Panasonic KX-P10xx family, and Smith Corona D300, with a document describing the installation process.

Amicus #10

This is an icon-driven demo, circulated to many dealers. It includes the sounds of an acoustic guitar, an alarm, banjo, calliope, car horn, claves, water drip, electric guitar, flute, harp arpegio, kickdrum, marimba, organ minor chord, people talking, pigs, pipe organ, rhodes piano, saxaphone, sitar, snare drum, steel drum, bells, vibrophone, violin, wailing guitar, horse whinny and whistle.

Amicus #11

C programs	
dirutil	intuition-based CLI replacement file manager
cpri	shows and adjusts priority of CLI processes
ps	shows info about CLI processes SE
vidtex	displays CompuServe RLE pictures SE
AmigaBasic programs	
pointered	pointer and sprite editor program
optimize	optimization example from A C article
calendar	large, animated calendar, diary and date book program
amortize	loan amortizations
brushtoBOB	converts small IFF brushes to AmigaBasic BOB objects
grids	draw and play waveforms
hilbert	draws hilbert curves
madlib	mad lib story generator
mailtalk	talking mailing list program
meadows.3D	3D graphics program, from Amazing Computing article
mousetrack	mouse tracking example in hires mode
slot	slot machine game
tictactoe	the game
switch	pachinko-like game
weird	makes strange sounds
Executable programs	
cp	unix-like copy command E
cls	clear screen program SE
diff	unix-like stream editor uses "diff" output to fix files
pm	chart recorder performances indicator
Assembler programs	
cls	screen clear and CLI arguments
Modula-2	
trails	moving-worm graphics demo
caseconvert	converts modula-2 keywords to uppercase
Forth	Breshehan circle algorithm example
Analyse	12 templates for the spreadsheet Analyze!
Other files	
	There are four programs here that read Commodore 64 picture files. They can translate Koala Pad, Doodle, Print Shop and News Room graphics to IFF format. Of course, getting the files from your C-64 to your Amiga is the hard part.

Using the Amiga as a Presentation Tool

During 1986, as part of the requirements of the maths/computer science degree at Swinburne, I had to give a one hour seminar on some area of management allied to mathematics. Being an ex chalkie, this did not worry me, however the weighting of 25% of the available marks meant that the presentation was very important.

Well, I thought, let's show this lot of IBM loving clowns what a real machine can do. I'll use my Amiga to do the presentation. DPSlide will do the trick.

Firstly, I researched the topic (always a good idea) and made up a rough draft of the areas I wished to cover. Out came Deluxe Paint and I made up a set of 'slides' that covered the main points of the talk. It was great to be able to specify text colour, size, font and positioning on the screen, and to draw graphs easily and neatly.

This is looking good, I mused, but something more spectacular is required. At the moment all I've got is a very expensive overhead projector. Then it struck me: I'll include some digitized images.

I searched through my PD disks and found some appropriate material. I also approached Maxwells and they kindly let me copy their digiview presentation slides. Combining both sources gave some magnificent (and appropriate) images. I also included some cartoons and drawings to break the tension of the talk and to point out some aspect in a humorous way.

The actual seminar went very well. The images (about 40) paced the talk very nicely, and the cartoons, digitized images etc. impressed the lecturers and other students enormously.

As a method of presentation this beats virtually any other method hands down. I would recommend it highly.

-- Mike Creek

Editorial

Well, most of you seem to have missed out on the premiere social event of the year. Yes, I'm talking about the **First Annual AUG Barbecue.** The weather was fine and everyone enjoyed themselves. About 20 AUG members and their friends/lovers/families managed to have a great time, without a single computer in sight!

If you look at this newsletter closely, you'll see that we've got a new address, **PO Box 48, Boronia, 3155** (In fact, you'd hardly have to look closely - the address is mentioned 10 times!). There are a number of reasons for the change.

When the group was born, there were no post office boxes available, so we "piggy-backed" on Ron Wail's personal P.O. box. Six months or so later, the box we had applied for in Boronia became available. We didn't do anything about it then, because it was still convenient to use Ron's. Then, a few months back, Ron moved from North Balwyn to Blackburn. This has made the mail a little hard to pick up, which has delayed our answers to your letters and orders.

The other problem which has forced the change is that the group has grown so large so quickly that we have been overwhelmed. We now have over 370 members, and it takes a great deal of time to answer all the letters, enquiries and disk orders. We apologise for the delays, and hope that the change of address will help us serve you faster. Instead of one person (ie Ron) emptying the box infrequently, and then having to do most of the work involved in disk copying and answering letters, etc., we are going to spread the workload out over more people.

A few people have asked why we don't copy the public domain disks at club meetings anymore. The problem is that no-one wants to miss the meetings! If someone with a two (or more) drive machine is prepared to bring their machine and copy disks at the meetings, then **please do!**

-- Peter Jetson

SOFTWARE ORDER FORM

Disk numbers :

Disks supplied by Amiga User Group @ $10	$
Disks supplied by member @ $2	$
Club Use Only Receipt #: Mailed on: / /	Total $

Mail to: **Amiga Users Group, PO Box 48, Boronia, 3155, Victoria.**

Member's Name:

Address:

Application for membership of The Amiga Users Group

Membership is $20 per year. Make cheques payable to The Amiga Users Group, and send to:

Amiga Users Group, PO Box 48, Boronia, 3155

Details this side are optional

Surname: _____

First name: _____ (no initials)

Address: _____

_____ Postcode: _____

Where did you hear about AUG: _____

How can AUG help you: _____

_____

_____

Signed: _____ Date: _____

In the event of my admission as a member, I agree to abide by the rules of the Association for the time being in force.

Phone number: _____ STD code: _____

Year of birth: _____

Occupation: _____

Interests: _____

_____

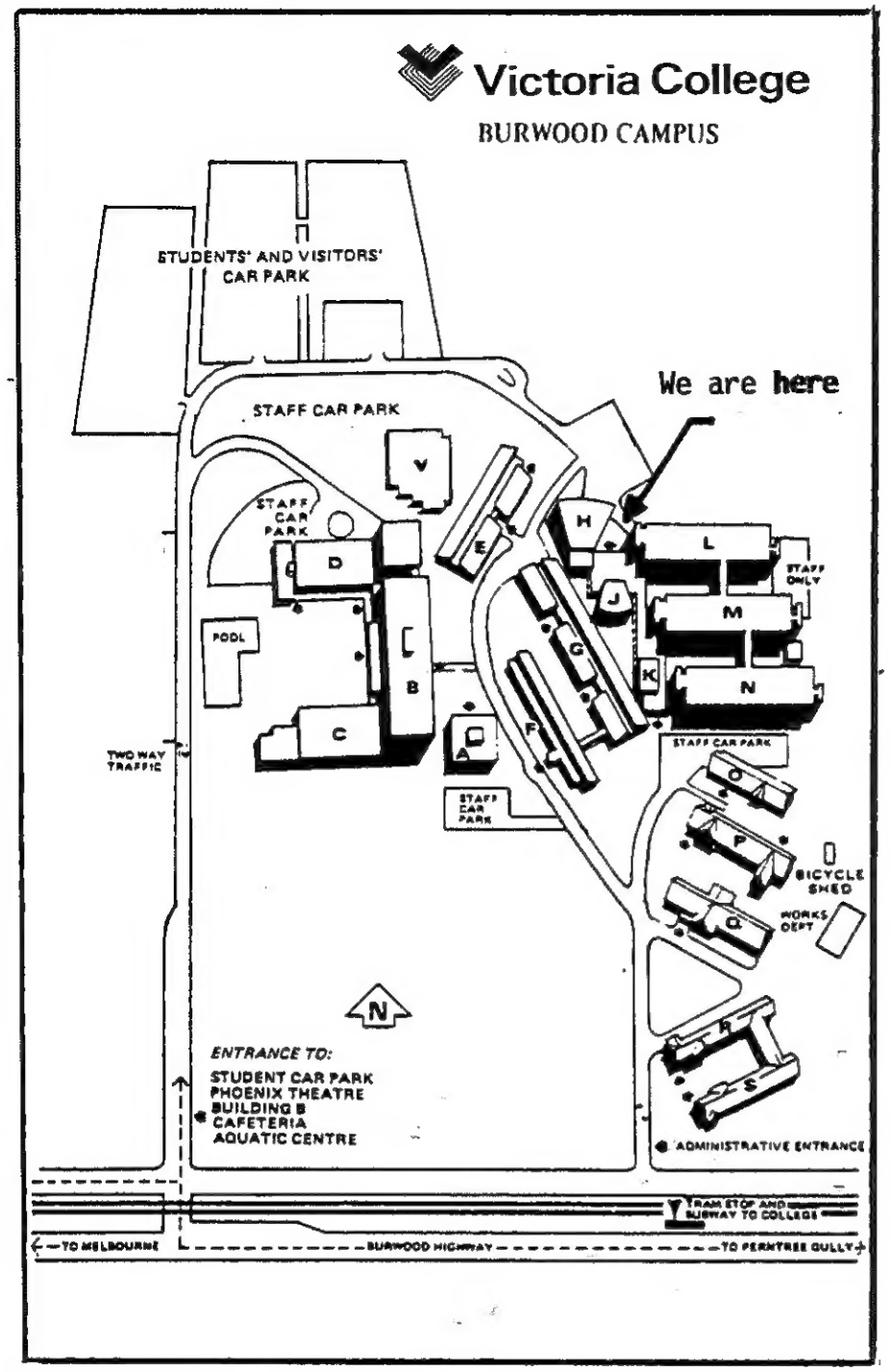
_____

Dealer's Name: _____

Dealer's Address: _____

_____ Postcode: _____

Are you happy with your dealer: _____



Where is Victoria College Burwood Campus?

New members and visitors sometimes have trouble locating our meeting place the first time. Victoria College is on the North side of Burwood Highway, Burwood, just East of Elgar Road. Coming from the City, turn left at the first set of traffic lights after Elgar Road. Follow the road around past the football oval, over three or four traffic bumps to the car parking areas near the netball courts. Further up the road, to the left, you'll find Lecture Theatre 2. That's us!

If you have a Melways, try Map 61 B5.

February 1987 Amiga Workbench

P.O. Box 48, Boronia, 3155, Victoria, Australia

AMIGA™ Users Group

Postage Paid
Caulfield
East 3145

3001

+ Member # 10297 Expires end of 11-87

Campbell Gray
RMIT Comm Eng Dept
Box 2476V
Melbourne

AMIGA Workbench
Registered by Australia Post
Publication No: VBG7930
If Undeliverable, return to:
PO Box 48, Boronia, 3155
Victoria, Australia